ps1

Other Formats
PDF

AUTHOR
zixuan Lan

PUBLISHED
October 5, 2024

1. **PS1:** Due Sat Oct 5 at 5:00PM Central. Worth 50 points.

We use (*) to indicate a problem that we think might be time consuming.

Steps to submit (5 points on PS1)

1. "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: zixuan lan
2. "I have uploaded the names of anyone I worked with on the problem set here" zixuan lan (1 point)
3. Late coins used this pset: **__** Late coins left after submission: I use late coins for this pset **__**
4. Knit your ps1.qmd to make ps1.pdf. The PDF should not be more than 25 pages. Use head() and re-size figures when appropriate.
5. Push ps1.qmd and ps1.pdf to your github repo. It is fine to use Github Desktop.
6. Submit ps1.pdf via Gradescope (4 points)
7. Tag your submission in Gradescope

```python
import altair as alt
from vega_datasets import data as vega_data
import pandas as pd
import time
import os
import matplotlib.pyplot as plt
```

Read in one percent sample (15 Points)

Q1

```python
starting_time = time.time()
df = pd.read_csv('file:///Users/lanzixuan/Desktop/parking_ticke
ending_time = time.time()

whole_time = ( ending_time - starting_time )
print(whole_time)
```

```
assert len(df) == 287458
```

```
0.6091880798339844

/var/folders/50/tht0jybj2dd592scvqr5ccgh0000gn/T/ipykernel_3982
/2850387733.py:2: DtypeWarning: Columns (7) have mixed types.
Specify dtype option on import or set low_memory=False.
  df =
pd.read_csv('file:///Users/lanzixuan/Desktop/parking_tickets_on
e_percent.csv')
```

Successfully add an assert statement which verifies that there are 287458 rows.

Q2

```
file_size = os.path.getsize('/Users/lanzixuan/Desktop/parking_t
full_size = file_size * 100
print(f'file_size: {file_size:.2f} MB')
print(f'full_size: {full_size:.2f} MB')
```

```
file_size: 79.78 MB
full_size: 7978.00 MB
```

The size of the full data set is 7978.00 MB

Q3

```
##Q3


def sorted(data, column_name):
    if data[column_name].is_monotonic_increasing:
        print(f" ascending order.")
        return True
    elif data[column_name].is_monotonic_decreasing:
        print(f" descending order.")
        return True
    else:
        print(f" not sorted.")
        return False

subset=df.iloc[0:500]
sorted(subset, 'issue_date')
```

```
 ascending order.

True
```

Cleaning the data and benchmarking (15 Points)

Q1

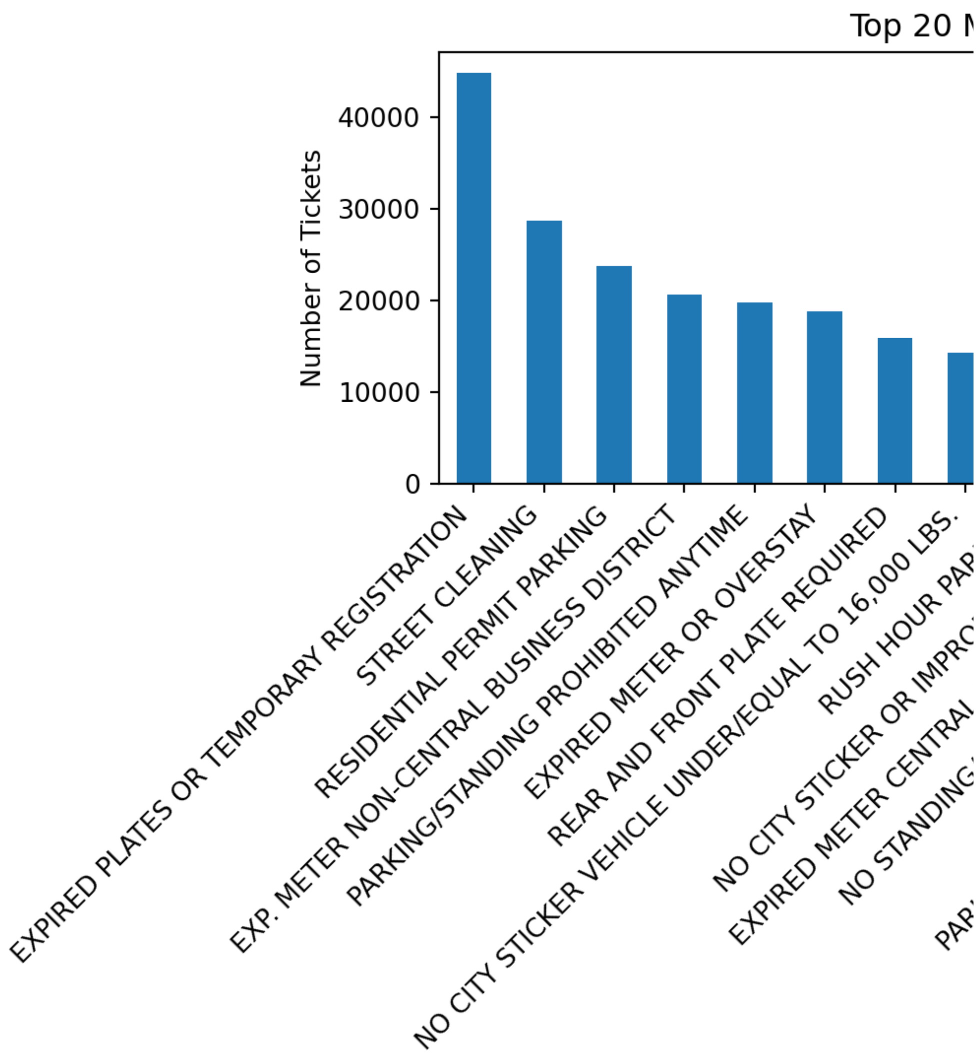
```
df['issue_date'] = pd.to_datetime(df['issue_date'])
tickets_for_2017 = df[df['issue_date'].dt.year == 2017]
tickets_for_2017_count = len(tickets_for_2017)
full_data_tickets_for_2017 = tickets_for_2017_count * 100
article_tickets_for_2017 = 3000000
difference = abs(full_data_tickets_for_2017 - article_tickets_fo
print(difference)
#It's meaningful
```

```
763600
```

```
##Q2
top20_violations = df['violation_description'].value_counts().he
print(top20_violations)
plt.figure(figsize=(10, 6))
top20_violations.plot(kind='bar')
plt.title('Top 20 Most Frequent Violation Types')
plt.xlabel('Violation Type')
plt.ylabel('Number of Tickets')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
violation_description
EXPIRED PLATES OR TEMPORARY REGISTRATION             44811
STREET CLEANING                                      28712
RESIDENTIAL PERMIT PARKING                           23683
EXP. METER NON-CENTRAL BUSINESS DISTRICT             20600
PARKING/STANDING PROHIBITED ANYTIME                  19753
EXPIRED METER OR OVERSTAY                            18756
REAR AND FRONT PLATE REQUIRED                        15829
NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 LBS.   14246
RUSH HOUR PARKING                                    11965
NO CITY STICKER OR IMPROPER DISPLAY                  10773
EXPIRED METER CENTRAL BUSINESS DISTRICT               9736
NO STANDING/PARKING TIME RESTRICTED                   8640
```

```
WITHIN 15' OF FIRE HYDRANT                          6104
PARK OR STAND IN BUS/TAXI/CARRIAGE STAND            6004
TRUCK,RV,BUS, OR TAXI RESIDENTIAL STREET            4789
STREET CLEANING OR SPECIAL EVENT                    3370
DOUBLE PARKING OR STANDING                          2904
EXPIRED PLATE OR TEMPORARY REGISTRATION             2720
STOP SIGN OR TRAFFIC SIGNAL                         2191
PARK OR BLOCK ALLEY                                 2050
Name: count, dtype: int64
```



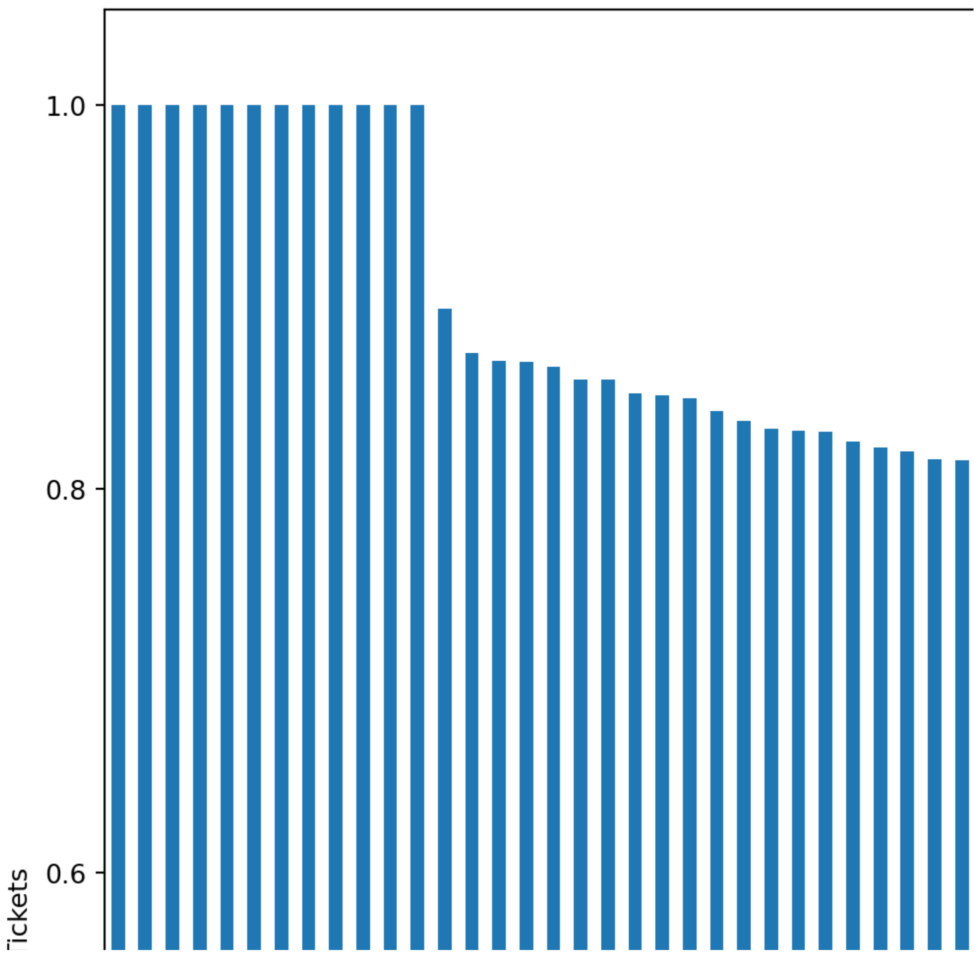
Visual Encoding (15 Points)

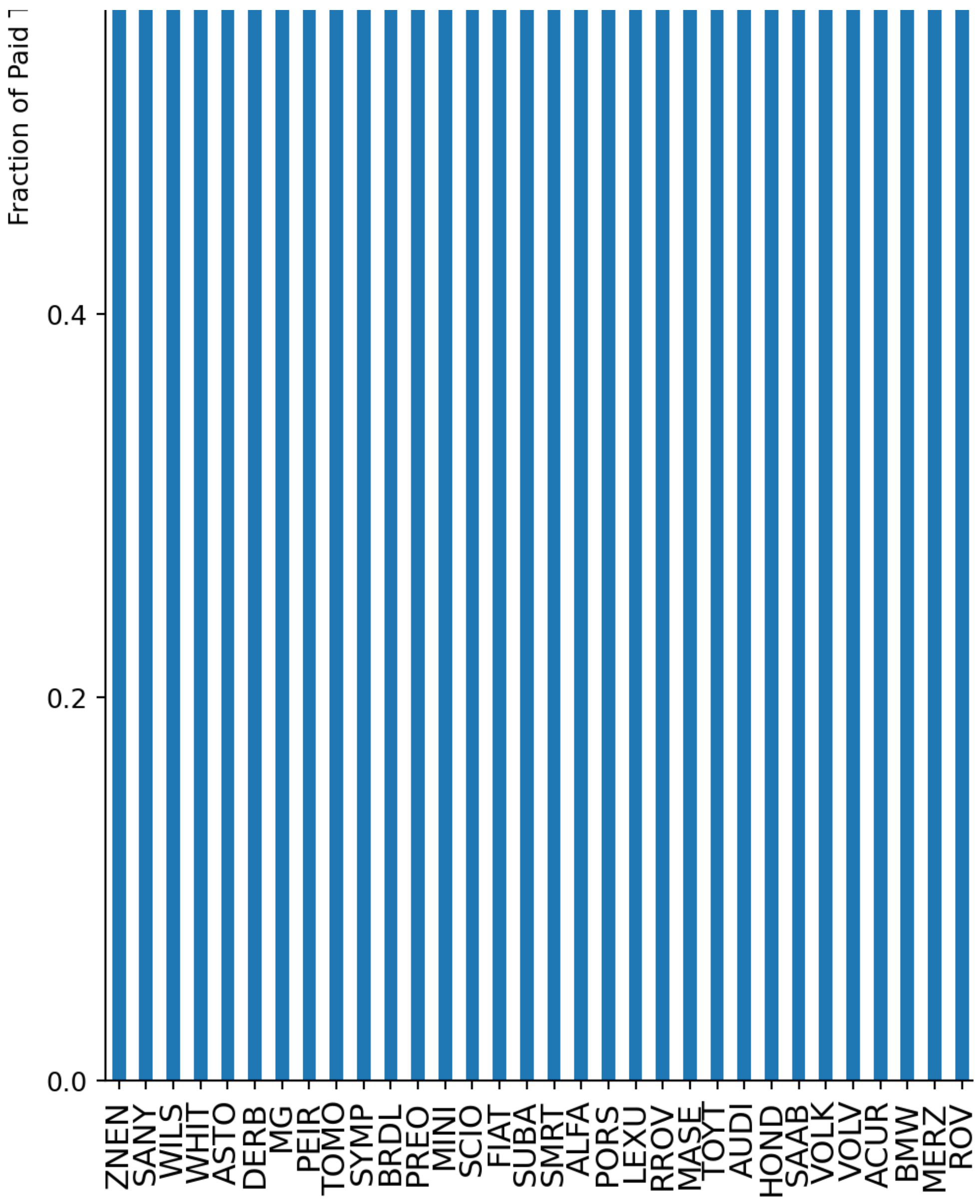
Question 1

Variable Name	Variable Type(s)
`ticket_number`	Nominal
`issue_date`	Temporal/Ordinal
`violation_location`	Nominal
`license_plate_number`	Nominal
`license_plate_state`	Nominal
`license_plate_type`	Nominal
`zipcode`	Nominal/Quantitative
`violation_code`	Nominal
`violation_description`	Nominal
`unit`	Nominal
`unit_description`	Nominal
`vehicle_make`	Nominal
`fine_level1_amount`	Quantitative
`fine_level2_amount`	Quantitative
`current_amount_due`	Quantitative
`total_payments`	Quantitative
`ticket_queue`	Nominal
`ticket_queue_date`	Temporal
`notice_level`	Nominal
`hearing_disposition`	Nominal
`notice_number`	Nominal
`officer`	Nominal
`address`	Nominal

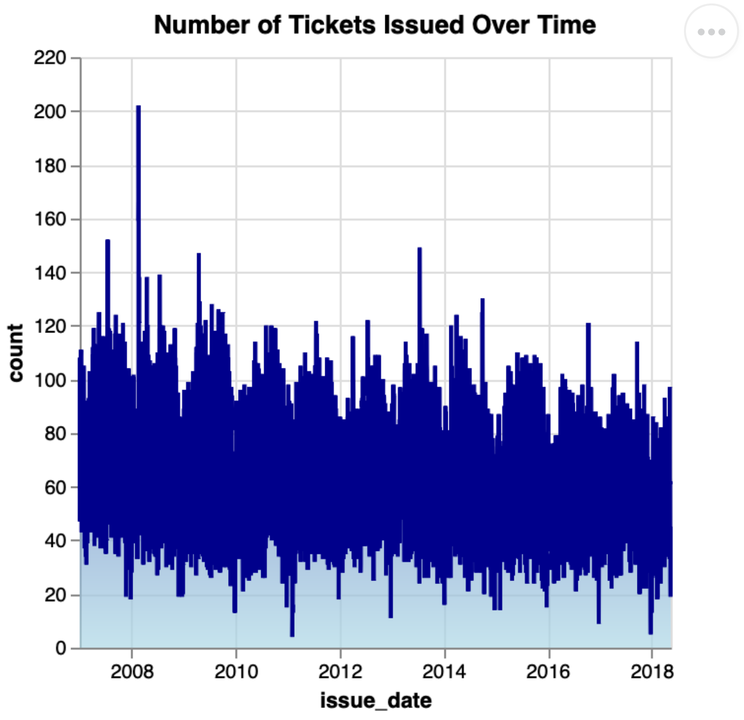
```
#Question 2
```

```
pd.set_option('display.max_columns', None)
df.head(20)

df['is_paid'] = df['current_amount_due'] == 0

df['Paid Status'] = df['ticket_queue'].map({'Paid': True, 'Defi


paid_fractions = df.groupby('vehicle_make')['Paid Status'].mean


plt.figure(figsize=(20, 12))
paid_fractions.sort_values(ascending=False).plot(kind='bar')
plt.title('Fraction of Paid Tickets by Vehicle Make')
plt.xlabel('Vehicle Make')
plt.ylabel('Fraction of Paid Tickets')
plt.xticks(rotation=90, fontsize=12)
plt.tight_layout()
plt.show()
#Maybe different brands of vehicles represent different income
```

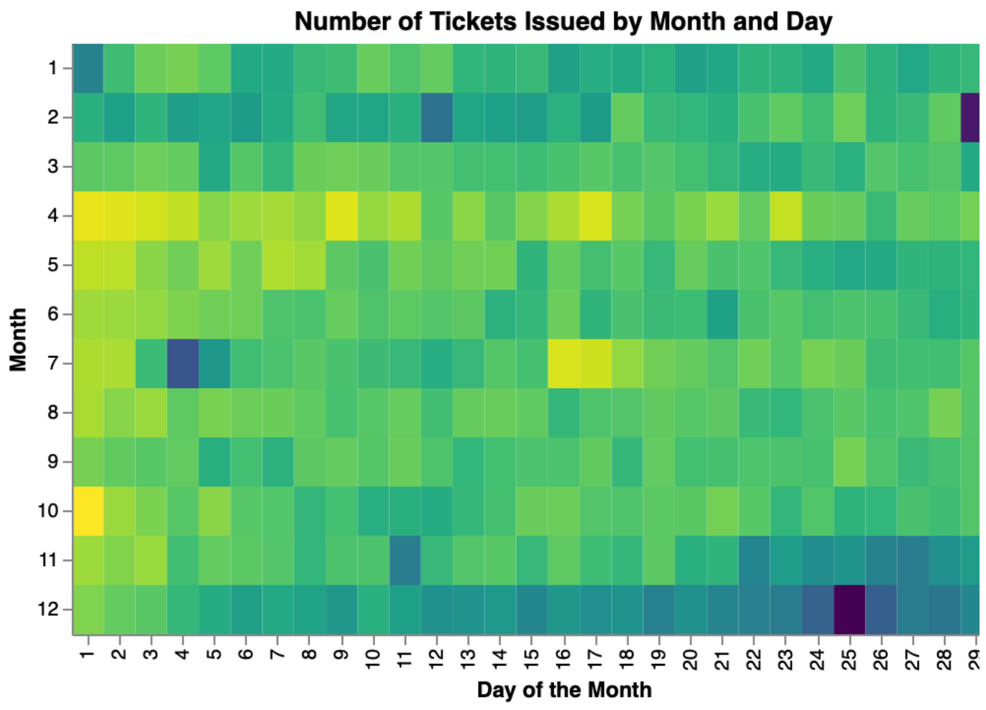




```
#Question 3
tickets_per_day = df.groupby(df['issue_date'].dt.date).size().re
tickets_per_day['issue_date'] = pd.to_datetime(tickets_per_day[

chart = alt.Chart(tickets_per_day).mark_area(
    line={'color':'darkblue'},
    color=alt.Gradient(
        gradient='linear',
        stops=[alt.GradientStop(color='lightblue', offset=0),
               alt.GradientStop(color='darkblue', offset=1)],
        x1=1,
        x2=1,
        y1=1,
        y2=0
    )
).encode(
    x='issue_date:T',
    y='count:Q'
).properties(
    title="Number of Tickets Issued Over Time"
)


chart
```

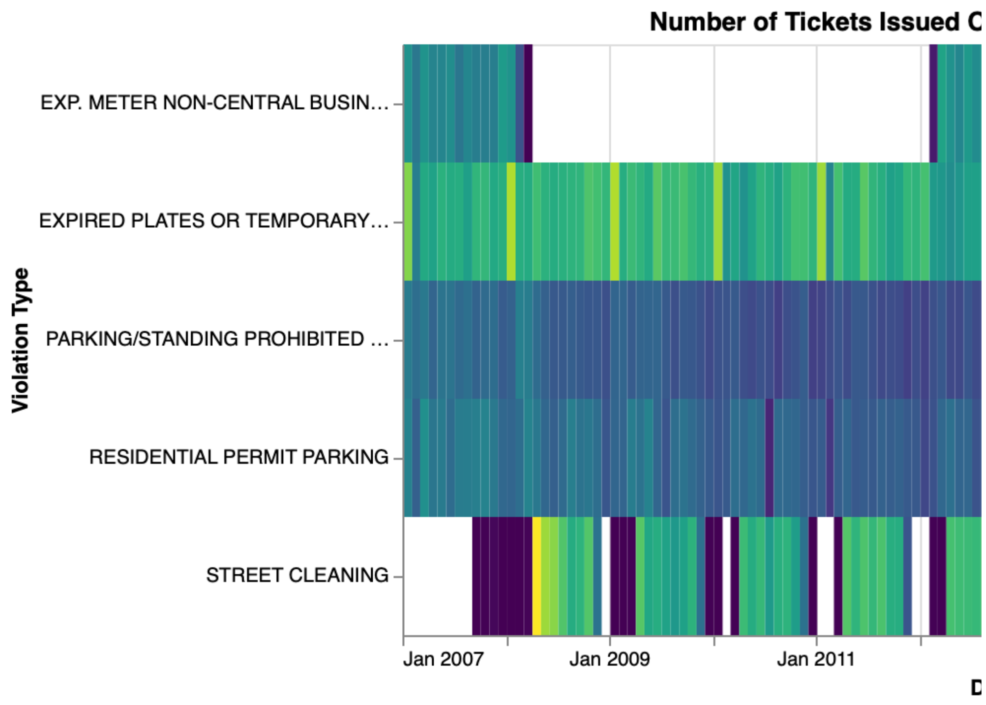


```
#Question 4
df['month'] = df['issue_date'].dt.month
df['day'] = df['issue_date'].dt.day


tickets_by_day = df.groupby(['month', 'day']).size().reset_inde


heatmap = alt.Chart(tickets_by_day).mark_rect().encode(
    x=alt.X('day:O', title='Day of the Month'),
    y=alt.Y('month:O', title='Month'),
    color=alt.Color('count:Q', scale=alt.Scale(scheme='viridis'
    tooltip=['month', 'day', 'count']
).properties(
    width=500,
    height=300,
    title='Number of Tickets Issued by Month and Day'
)


heatmap
```



```
##Question 5
df['issue_date'] = pd.to_datetime(df['issue_date'])

top_5_violations = df['violation_description'].value_counts().he

df_top_violations = df[df['violation_description'].isin(top_5_v

tickets_by_time_and_violation = df_top_violations.groupby([pd.G

tickets_by_time_and_violation['issue_date'] = pd.to_datetime(ti

alt.data_transformers.disable_max_rows()

lasagna_plot = alt.Chart(tickets_by_time_and_violation).mark_re
    x=alt.X('yearmonth(issue_date):T', title='Date'),
    y=alt.Y('violation_description:N', title='Violation Type'),
    color=alt.Color('count:Q', scale=alt.Scale(scheme='viridis'
    tooltip=['issue_date', 'violation_description', 'count']
).properties(
    width=600,
    height=300,
    title='Number of Tickets Issued Over Time for Top 5 Violati
)


lasagna_plot
```

```
/var/folders/50/tht0jybj2dd592scvqr5ccgh0000gn/T/ipykernel_3982
/843889953.py:8: FutureWarning: 'M' is deprecated and will be
removed in a future version, please use 'ME' instead.
  tickets_by_time_and_violation =
df_top_violations.groupby([pd.Grouper(key='issue_date',
freq='M'),
'violation_description']).size().reset_index(name='count')
```



Question 6

#1. Filled Step Chart #Pros: #It clearly shows the overall trend of time over time, especially suitable for displaying time series data. #It is easy to see which time periods have more tickets issued. #Filled areas make trends more intuitive. #Cons: #It can only display data of one dimension (the total number of tickets), and cannot display other dimensions (such as violation type, location, etc.) at the same time. #If the time span is large, details may be obscured. #2. Heatmap #Pros: #Heatmaps are very suitable for showing relationships between multiple dimensions, such as the distribution of tickets by day and month. #Color coding makes it easy to see high-frequency and low-frequency dates at a glance, and you can quickly see the peak period of issuing tickets. #Good for exploring cyclical or seasonal patterns in data. #Cons: #It is difficult to show continuous trends over time (unlike line charts or step charts). #If the color difference is not large, it may be difficult to distinguish between high-frequency and low-frequency areas. #Heatmaps are not suitable for showing very fine-grained trends or changes. #3. Lasagna Plot #Advantages: #It can show the trend of multiple categories (violation types) over time at the same time, which is suitable for comparing the time changes of different categories. #Color coding intuitively shows the number of tickets in each category, making it easy to observe the peak period of each type of violation. #Suitable for analyzing the relationship between multiple variables, especially time series data. #Disadvantages: #When there are too many violation types, the labels on the Y axis may become very crowded, affecting readability. #As with heat maps, the choice of color is very critical. If the color contrast is not enough, it may be difficult to distinguish the difference in the number of different tickets. #For large amounts of data, the graphics may appear complicated and it is not easy to see the overall trend at a glance.

` ## Question 7 #Best choice: Filled Step Chart #Reason: #Clearly show changes over time: #Filled step charts can clearly show trends and fluctuations in time series. If the issuance of tickets is very concentrated or fluctuates greatly in certain time periods, step charts can clearly show these peaks and valleys. #By filling in the area, you can easily see which period has more tickets issued and which period has relatively few. Any uneven enforcement behavior (such as a lot of tickets in certain time periods) will show up as obvious fluctuations or regional differences in the chart. #Suitable for time series data:

#From the time dimension, step charts are the most intuitive way to show time series. #Step charts show the number of tickets in each time period through a continuous curve, which is very helpful for explaining uneven enforcement.